PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK VISUALISASI HASIL PEMINDAIAN OPTIS TRIANGULAR DENGAN PEMANGKASAN OBJEK TIGA DIMENSI BERBASIS PARTISI OCTREE

Rully A. Hendrawan – Rully Soelaiman
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Email: eraha@inf.its-sby.edu

ABSTRAK

Perkembangan mesin pemindai tiga dimensi memungkinkan perngarsipan substansi fisik dari dokumen tiga dimensi. Tentunya agar data hasil pengarsipan ini dapat dinikmati oleh banyak orang, maka diperlukan sebuah media untuk mempublikasikan arsip-arsip tersebut. Komputer personal yang ada saat ini adalah salah satu alternatif yang memungkinkan.

Arsip-arsip digital tersebut pada saat ini umumnya adalah sebuah model 3-D dalam bentuk kumpulan mesh (permukaan yang dibangun dari poligon) yang padat poligon, bentuk geometri ini disimpan dalam suatu file berformat PLY. File ini nantinya akan dibaca dan divisualisasikan dengan bantuan OpenGL. Untuk memperlancar pengontrolan model dilakukan pemisakan antara proses visualisasi ketika sebuah model sedang dikontrol dan setelah selesai. Bentuk sederhana dari model tersebut ditampilkan ketika model sedang dikontrol, setelah selesai baru digambar bentuk asli dari model tersebut. Selain itu dilakukan juga pemangkasan bagian-bagian yang berada diluar batas pandang berbasis sel dengan menggunakan struktur data Octree untuk mempartisi ruang.

Perangkat lunak ini dapat memvisualisasikan struktur geometri dari sebuah model tanpa mempedulikan substansi permukaannya. Ketika model hanya tampak sebagian saja, proses visualisasi akan lebih cepat karena dilakukan pemangkasan pada bagian yang tidak tampak. Partisi ruang dengan menggunakan struktur data Octree mengurangi kompleksitas pada saat proses pemangkasan karena pengecekan terhadap batas pandang bukan dilakukan per poligon namun per partisi secara hirarki.

Kata Kunci : *Visualisasi 3D, Mesh Padat Poligon, Render, Octree.*

1. PENDAHULUAN

Meskipun metodologi-metodologi yang dibutuhkan untuk membuat dan mengatur arsip digital dua-dimensi telah semakin dewasa dalam sepuluh tahun terakhir, perpindahan dari dua menuju tiga dimensi menghadapi beberapa masalah. Masalah-masalah ini muncul dari kedua sisi skala dan substansi, dan meliputi segala aspek pengarsipan digital: penyimpanan, pengindeksan, pencarian, distribusi, penampilan, dan perlindungan dari pembajakan. Masalah yang akan dibahas disini yaitu bagaimana memvisualisasikan arsip-arsip tersebut dengan memanfaatkan komputer personal yang ada pada saat ini.

Tujuan pembuatan Tugas Akhir ini adalah untuk membuat sebuah perangkat lunak yang dapat menampilkan dan memanipulasi tampilan proyeksi model 3D yang dibangun dari poligon yang luar biasa banyaknya. Tanpa adanya pemacu grafis atau kartu grafis yang sangat bagus pun software ini dapat memvisualisasikan objek tersebut dan dapat dikontrol dengan lancar.

2. OPENGL

OpenGL [1] adalah sebuah sebuah pustaka untuk antarmuka antara perangkat lunak dengan perangkat keras grafis (pemacu aplikasi 3D). Antarmuka ini terdiri dari sekitar 120 perintah yang berbeda, dimana digunakan untuk menspesifikasikan objek-objek dan operasi-operasi yang diperlukan untuk menghasilkan aplikasi-aplikasi 3 dimensi yang interaktif. Tujuan utama dari OpenGL adalah melakukan render objek-objek 2 dimensi dan 3 dimensi menjadi sebuah *frame-buffer*. *Frame-buffer* adalah sebuah informasi dalam memori yang berisikan tentang gambar hasil *render*. Objek-objek tersebut digambarkan sebagai serangkaian verteks-verteks (mendefinisikan objek-objek geometri) atau piksel-piksel (yang mendefinisikan citra). OpenGL melakukan beberapa langkah-langkah pemrosesan data untuk mengkonversi data tersebut menjadi piksel-piksel untuk membentuk citra akhir yang diinginkan di dalam *frame-buffer*. Keseluruhan alur proses dalam OpenGL ini seringkali disebut *OpenGL rendering*

pipeline, dimana "*pipeline*" berarti sekumpulan proses yang menjadikan hasil sebuah proses sebagai masukan untuk proses lainnya.

OpenGL didesain sebagai sebuah antarmuka yang efisien dan independen terhadap perangkat keras untuk diimplementasikan pada platform-platform perangkat keras yang berbeda-beda. Untuk mendapatkan kualitas tersebut, tidak terdapat perintah-perintah untuk menampilkan *window/dialog* atau untuk mendapatkan masukan dari penguna, di dalam OpenGL; namun harus bekerja melalui sistem *window* apapun untuk mengontrol perangkat keras khusus yang digunakan. OpenGL tidak menyediakan perintah-perintah tingkat tinggi untuk menggambarkan model-model dari objek 3 dimensi. Dengan memanfaatkan OpenGL, maka untuk membuat model yang diinginkan harus dibangun dari sekumpulan kecil dari bentuk geometrik primitif, antara lain titik-titik, garis-garis, dan poligon-poligon.

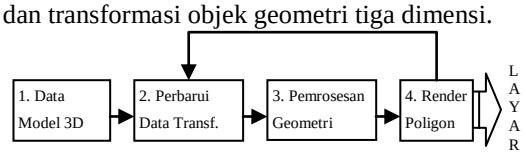
3. WXWIDGETS

wxWidgets adalah sebuah pustaka berbasis kelas yang memungkinkan untuk meng-compile program-program grafis C++ dalam beberapa platform yang berbeda. wxWidgets mendefinisikan API yang umum untuk semua platform, namun dengan menggunakan native graphical user interface (GUI bawaan sistem operasi) pada tiap-tiap platform, sehingga program akan tampak familiar dilihat oleh pengguna. wxWindows adalah sebuah framework C++ yang menyediakan GUI dan fasilitas-fasilitas lainnya pada lebih dari satu platform. Dimana versi 2 mendukung seluruh versi-versi desktop dari MS Windows, Unix dengan GTK+, Unix dengan Motif, dan MacOS.

4. ARSITEKTUR SISTEM

Perangkat lunak yang dibangun adalah berupa aplikasi berbasis *desktop*. Aplikasi ini nantinya dapat membaca format arsip tertentu yaitu file PLY dan memvisualisasikannya ke layar monitor.

Visualisasi yang dimaksudkan dalam hal ini mencakup keseluruhan dari struktur-struktur, fungsi-fungsi, dan algoritma-algoritma yang digunakan untuk visualisasi, yakni setelah beberapa perhitungan dan transformasi objek geometri tiga dimensi.



Gambar 1. Siklus Dasar Proses Visualisasi

Bagian-bagian utama dari visualisasi tiga dimensi yaitu:

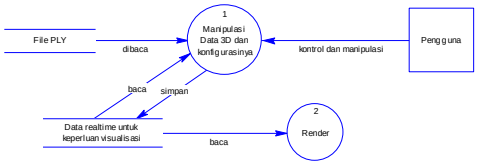
1. Data model didapatkan dengan membaca data objek 3D dari file PLY [2], yang kemudian disimpan ke dalam data proses. Data ini kemudian di partisi menggunakan struktur data Octree [5,6,7]
2. Apabila pengguna melakukan pengontrolan terhadap objek 3D maka didapatkan matriks transformasi yang bersesuaian.
3. Transformasi dilakukan untuk menempatkan objek-objek di dalam ruang dunia. Pemrosesan geometri yang dilakukan adalah pemangkasan lingkungan 3D terhadap batas pandang secara hirarki berbasis partisi Octree yang telah dibentuk pada tahap 1.
4. Informasi lingkungan 3D kemudian dilanjutkan ke OpenGL untuk di-render ke layar.
5. Proses terus berulang menunggu interaksi dari pengguna sehingga visualisasi akan terus diperbarui ketika didapatkan masukan.

Fungsi-fungsi tersebut diatas tergabung dalam rangka-kerja OpenGL. Dalam pemanfaatan OpenGL ini struktur program akan dibuat sedemikian rupa sehingga memenuhi "*OpenGL Rendering Pipeline*". Struktur program OpenGL ini dibagi menjadi beberapa bagian yang berbeda, antara lain:

- Fungsi Init: digunakan OpenGL dan untuk inisialisasi matriks-matriks model, *view* dan proyeksi. Kita dapat menaruh semua operasi-operasi inisialisasi yang diperlukan di dalam fungsi ini.
- Fungsi Resize: dipanggil tiap saat pengguna memulai program dan merubah resolusi output-window. Hal ini diperlukan untuk mengkomunikasikan *viewport* yang baru ke OpenGL [1].
- Fungsi Input: dipanggil tiap saat pengguna menekan tombol mouse atau keyboard.
- Siklus Utama: sebuah perulangan tak terhingga, dimana memanggil semua fungsi-fungsi untuk tiap frame. Pada bagian ini dilakukan proses pengambilan dan pembentukan data dunia, data transformasi dan pemrosesan geometri. Pemrosesan geometri yang dilakukan salah satunya adalah pembentukan partisi Octree [5,6,7].
- Fungsi Drawing: menghapus semua *buffer-buffer* (stensil, warna dan kedalaman). Semua transformasi model, *view* dan proyeksi dilakukan dan *scene* digambar. Terakhir dua *buffer* ditukar untuk menaruh gambar pada layar (metode double buffering) [1].

5. PERANCANGAN APLIKASI

Aplikasi dibagi menjadi dua proses utama, yaitu manipulasi data geometri 3D (yang berupa verteks-verteks dan poligon-poligon serta data-data manipulasi yang berhubungan dengannya), dan proses pemetaan data-data tersebut menjadi bentuk visual 2D yang dapat dilihat pada monitor (juga biasa disebut rendering). Proses-proses ini merupakan serangkaian proses yang dijalankan secara berurutan.



Gambar 2. Proses Visualisasi

Proses diawali dengan memasukkan data yaitu file PLY [2]. Dimana data-data tersebut kemudian dikontrol dan dimanipulasi oleh pengguna dengan bantuan prosedur-prosedur yang sudah ada untuk kemudian disimpan ke memori sebagai data proses untuk keperluan render.

6. TRANSFORMASI ANTAR RUANG

Mendifinisikan ruang yang berbeda dalam 3D sangatlah penting. Salah satu kesepakatan adalah membagi ruang menjadi empat yaitu ruang objek, ruang dunia, ruang pandangan/view, dan ruang layar.

- Ruang Objek (ruang 3D)
- Sistem koordinat ini adalah lokal terhadap sebuah objek, dimana sebuah objek terdiri atas kumpulan poligon. Jika terdapat banyak objek yang sama pada lokasi yang berbeda dalam dunia, ruang objek sangatlah diperlukan.
- Ruang Dunia (ruang 3D)
- Sistem koordinat ini adalah yang paling penting karena bersifat global. Disinilah seluruh objek ditempatkan. Dimana perhitungan fisika, pergerakan, deteksi tumbukan, dan juga komputasi pencahayaan dilakukan.
- Ruang Pandangan (ruang 3D)
- Koordinat sistem ini bersifat relatif terhadap kamera. Objek-objek dalam ruang dunia ditransformasikan ke ruang pandangan / *view space* untuk mengetahui apakah nampak dalam layar.
- Ruang Layar (ruang 2D)
- Ruang Layar adalah koordinat homogen yang merupakan representasi dari layar. Koordinat bukanlah lokasi piksel. *Viewport* memetakan ruang pandang ke ruang layar, dimana titik pusat (origin) dari ruang layar adalah ditengah layar (untuk proyeksi perspektif)
-

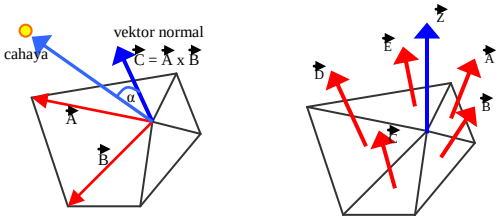
Untuk memindahkan koordinat verteks dari ruang satu ke ruang lainnya diperlukan pemetaan. Pemetaan ini diwujudkan sebagai matriks transformasi. Karena terdapat empat ruang maka dibutuhkan tiga matriks transformasi yaitu:

- MATRIKS MODEL (MM)
  dari ruang model (koordinat lokal) ke ruang dunia (koordinat global)
- MATRIKS VIEW (MV)
  dari ruang dunia ke ruang pandang/view
- MATRIKS PROYEKSI (MP)
  Dari ruang pandang ke ruang layar

Perlu diingat bahwa transformasi adalah perkalian matriks dimana M1 x M2 tidak sama dengan M2 x M1. Pada OpenGL, matriks model dan matriks view digabung menjadi sebuah matriks yaitu matriks transformasi MODELVIEW.

7. VEKTOR NORMAL

Vektor normal menentukan bentuk arsiran model, terdapat dua macam yaitu arsir datar dan halus.



normal sisi normal verteks

Gambar 3. Dua Macam Vektor Normal

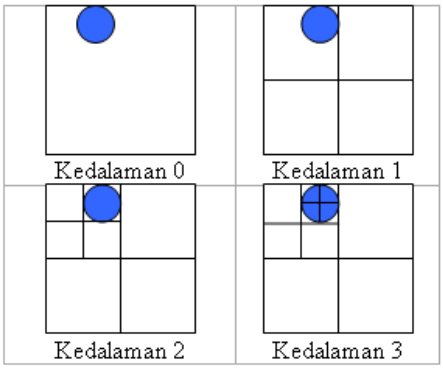
Arsir datar diperoleh dengan menghitung vektor normal sebagai perkalian vektor (*cross product*) yang kemudian dinormalisasikan. Sedangkan arsir halus didapatkan dengan merata-rata semua vektor sisi yang terhubung verteks tersebut yaitu dengan menjumlahkan vektor normal dari sisi sekitar verteks kemudian menormalisasinya.

8. PARTISI RUANG METODE OCTREE

Pada dasarnya metode Octree membagi sebuah ruang menjadi 8 bagian, setiap bagiannya akan terus-menerus dibagi menjadi 8 secara rekursif selama memenuhi dua syarat berikut:

6. Sebuah hasil partisi mengandung sejumlah objek pada batas minimum tertentu
7. Hirarki Octree mencapai batas kedalaman tertentu

Kedua syarat diatas mengakibatkan munculnya dua buah parameter dalam pembentukan Octree yaitu jumlah minimum objek pada sebuah bagian partisi dan batas kedalaman hirarki.



Gambar 4. Pembentukan Octree Tampak Atas

Sebuah partisi didefinisikan sebagai suatu area pada lokasi tertentu dengan radius tertentu. Partisi tersebut memiliki batas dalam dua bentuk yaitu kubus

dan bola. Apabila pertisi tersebut masih dapat dibagi, maka akan disimpan penunjuk 8 hasil pembagiannya (node anak). Apabila partisi tersebut berada dalam level terbawah (tidak dapat dibagi lagi), maka partisi tersebut juga berisi daftar seluruh segitiga yang terkandung didalamnya.

Perlu diperhatikan bahwa partisi ini dilakukan pada ruang model dengan tujuan agar nantinya pengecekan terhadap batas pandang dilakukan berbasis per bagian partisi bukan per poligon. Pengecekan berbasis poligon kurang efektif karena poligon yang ada sangatlah banyak jumlahnya.

9. PEMANGKASAN BAGIAN YANG DILUAR BATAS PANDANG

Batas pandang dalam dunia 3D didefinisikan dalam 3 bentuk, yaitu:

- Piramida terpotong (6 bidang potong) [3]
- Persamaan sebuah bidang adalah Ax+By+Cz+D = 0, sehingga dapat direpresentasikan sebagai matriks 1x4 untuk menyimpan nilai A,B,C dan D.
- Kerucut (posisi, arah, FOV)
- Bola (posisi, radius)

Pemangkasan dilakukan dengan pengecekan secara berlapis mulai dari kompleksitas terkecil sampai terbesar. Terdapat 4 tahap, yaitu:

- Pertama-tama dilakukan pengecekan batas partisi berbentuk bola (RB) terhadap batas pandang (RV) berbentuk bola dimana: jika jarak antara posisi partisi dengan posisi titik pandang kurang dari RB+RV maka kedua batas bersinggungan, sehingga partisi tersebut sebagian berada dalam batas pandang.
- Selanjutnya pengecekan dilakukan pada batas partisi berbentuk bola terhadap batas pandang kerucut [4].
- Apabila pada tahap ini partisi masih terbukti berada dalam batas pandang, maka dilanjutkan dengan pengecekan batas pandang berbentuk bola terhadap 6 bidang potong.
- Jika masih lolos maka dilakukan tahap terakhir yaitu pengecekan titik-titik sudut batas partisi berbentuk kubus terhadap 6 bidang potong. Jarak antara sebuah titik dengan bidang potong dapat dihitung dengan menggunakan perkalian skalar antara vektor posisi titik dengan vektor normal dari bidang.

Pada suatu tahap pengecekan, jika sebuah partisi terbukti seluruhnya berada di dalam atau di luar batas pandang maka pengecekan selanjutnya dan juga terhadap bagian-bagian bawah (node anak) tidak perlu dilakukan.

Kelebihan dari metode ini adalah pengecekan tidak dilakukan untuk seluruh verteks pada poligon. Namun terdapat kekurangan yaitu pembentukan struktur Octree cukup mahal.

10. UJI COBA DAN EVALUASI

Uji coba terhadap perangkat lunak dilakukan dengan menggunakan arsip model HORSE.PLY, BUNNY.PLY, DRAGON.PLY, HAPPY.PLY dan BLADE.PLY. Dari masing-masing model tersebut akan diuji coba mengetahui waktu yang dibutuhkan dalam membaca model itu. Uji coba yang akan dilakukan adalah sebagai berikut :

8. Uji coba kebenaran
9. Uji coba dengan model yang berbeda
10. Uji coba dengan peningkatan jumlah model
11. Uji coba dengan peningkatan jumlah verteks dan poligon
12. Uji coba pada komputer yang berbeda

Perangkat keras yang digunakan dalam uji coba ini adalah sebuah komputer personal dengan spesifikasi standar tahun 2003. Spesifikasi dari komputer yang digunakan adalah prosesor AMD Athlon XP 2000+ 1,67 GHz, memori sebesar 256 MB, VGA Card G-Force MX 440 64MB. Sistem operasi yang digunakan adalah Windows XP Professional.

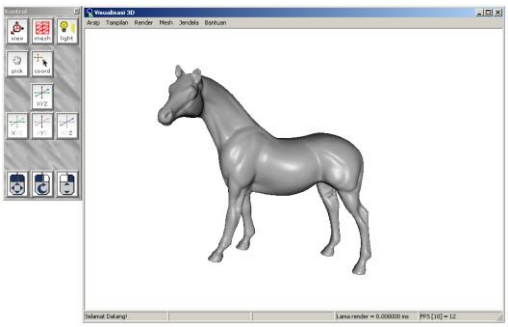
10.1. UJI COBA KEBENARAN

Proses pelaksanaan uji coba kebenaran perangkat lunak dibagi menjadi beberapa bagian, yang akan dijelaskan sebagai berikut :

Uji coba mode penggambaran

Proses pelaksanaan uji coba kebenaran perangkat lunak dilakukan dengan mencoba berbagai macam mode penggambaran objek yang tersedia. Mulai dari yang berbasis poligon, kemudian rangka, titik dan dua-resolusi.

Seperti yang terlihat pada gambar berikut, mode arsir poligon halus menghasilkan visualisasi yang terbaik namun juga yang menghabiskan waktu paling lama.



Gambar 5. Hasil Arsir Halus Mengkilat

Selanjutnya dicoba mode pengarsiran datar yang menggunakan vektor normal sisi, sekaligus membandingkan apabila opsi mengkilat di non-aktikan. Perhatikan gambar 5, pada arsiran datar tampak sekali siku-siku dari segitiga-segitiga yang membangun permukaan.

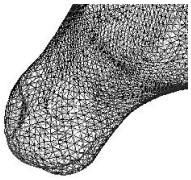

Datar, Tidak mengkilat

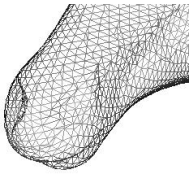

Datar, Mengkilat

Gambar 6. Hasil Arsir Datar

Bagian kanan gambar 5, adalah hasil objek diarsir datar namun dengan mengaktifkan permukaan mengkilat dengan mengubah material dari poligon-poligon yang membangunnya. Tampak sekali pada daerah tertentu yang semakin tegak lurus dengan arah cahaya, poligon-poligon tersebut semakin bersifat memantulkan cahaya yang diterimanya.

Uji coba selanjutnya adalah mencoba mode rangka dan titik yang dapat dilihat hasilnya sebagai berikut:


Rangka Utuh

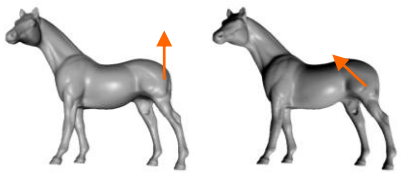

Rangka dengan Permukaan Tersembunyi


Semua Verteks Digambar


Titik dengan Permukaan Tersembunyi

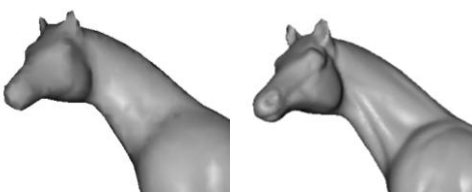
Gambar 7. Hasil mode Rangka dan Titik

Dengan memindah sumber cahaya tampak bahwa bayangan model berpindah letaknya.



Gambar 8. Hasil Pemindahan Sumber Cahaya

Dengan mode dua-resolusi, model digambar dengan menggunakan resolusi yang rendah ketika sedang dikontrol. Dan apabila telah selesai diperbaiki kualitasnya dengan menggambar ulang menggunakan resolusi aslinya.



Gambar 9. Mode Gambar Dua-Resolusi

1. Uji Coba dengan Model Berbeda

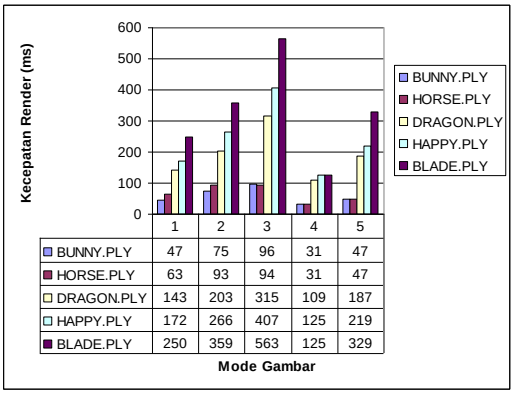
Uji coba ini dilakukan untuk mengetahui waktu yang dibutuhkan pada saat me-render model yang berbeda. Uji coba ini dilakukan dengan menggunakan lima model yang memiliki jumlah poligon yang berbeda serta ukuran masing-masing arsip yang berbeda.

Tabel 1. Ringkasan Hasil Uji Coba Kecepatan Render pada Lima Model

Model	Jml Poligon	Kecepatan pada Mode Gambar (ms)				
		Arsir	Rangka	Rangka (PT)	Titik	Titik (PT)
BUNNY	69451	47	75	86	31	47
HORSE	96966	47	78	141	31	47
DRAGON	871414	141	203	312	109	187
HAPPY	1087716	172	266	407	125	219
BLADE	1765388	250	359	563	125	329

Uji coba dilakukan pada kelima model dimana mode gambar yang diuji adalah arsiran penuh, rangka, rangka tanpa permukaan tersembunyi, titik, dan titik tanpa permukaan tersembunyi. Uji coba dilakukan untuk mendapatkan waktu render yang diperlukan untuk tampak depan, kanan, belakang, kiri, atas dan bawah. Kemudian dihitung untuk mendapatkan waktu rata-rata dari keseluruhan sudut pandang dalam satuan milisekon (ms). Uji coba dilakukan dengan mengaktifkan pencahayaan dan menggunakan resolusi yang asli.

Dari informasi diatas diperoleh grafik kecepatan proses render terhadap model uji coba pada mode gambar yang berbeda. Tampak pada gambar 5.7 bahwa mode gambar yang paling cepat adalah mode gambar titik (4) dimana hanya digambar verteks-verteksnya saja. Kemudian disusul oleh mode gambar arsir (1), titik tanpa permukaan tersembunyi (5), rangka (2), dan rangka tanpa permukaan tersembunyi (3).



Gambar 10. Grafik Kecepatan Proses Render pada Lima Model

Mode gambar rangka dengan menghilangkan permukaan tersembunyi menghabiskan waktu paling lama disebabkan oleh dua hal, yaitu:

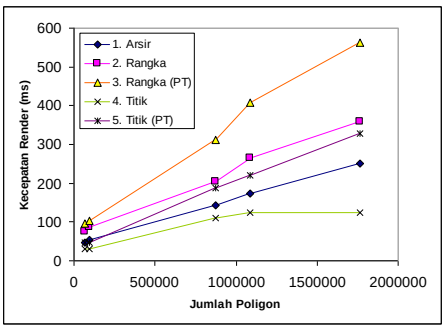
13. Mode gambar rangka pada model dengan jumlah poligon yang sangat banyak menghabiskan waktu lebih lama dibanding mode arsir karena

seluruh poligon harus diproses, baik yang tampak maupun tidak.

14. Untuk menghilangkan permukaan tersembunyi diperlukan proses tambahan.

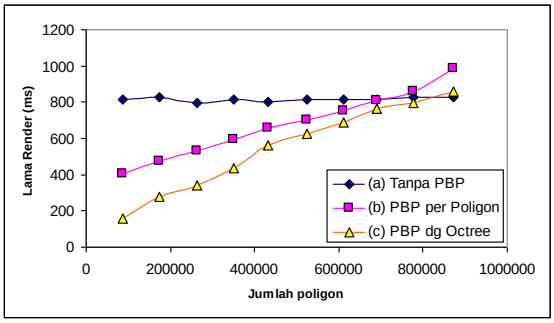
2. Uji Coba Peningkatan Jumlah Poligon

Uji coba kali ini dengan untuk menunjukkan pengaruh jumlah poligon terhadap kecepatan proses render.



Gambar 11. Grafik Kecepatan Proses Render terhadap Jumlah Poligon

Uji coba peningkatan jumlah poligon selanjutnya menggunakan model DRAGON.PLY pada mode gambar rangka dan pencahayaan diaktifkan. Pertama-tama ditampilkan sekitar sepersepuluh dari seluruh poligon yang ada yaitu sebanyak 87.605 poligon dari total 871.414, waktu yang dibutuhkan untuk menampilkannya adalah 144 milisekon. Demikian dilanjutkan dua per sepuluhnya, tiga per sepuluh dan seterusnya sampai keseluruhan model ditampilkan. Dari grafik tampak perbandingan kecepatan ketika tanpa pemangkasan terhadap batas pandang (a), dilakukan pemangkasan batas pandang dengan metode per poligon (b), dan pemangkasan batas pandang berbasis partisi octree (c).



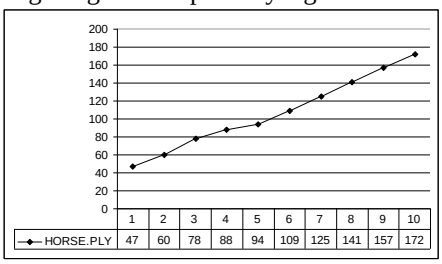
Gambar 12. Grafik Kecepatan Render terhadap Jumlah Poligon yang Tampak

Dari hasil uji coba ini diketahui bahwa yang sangat berpengaruh terhadap kecepatan render adalah jumlah poligon yang tampak. Dan dengan menggunakan Octree, ketika keseluruhan objek tampak, tidak terjadi pengecekan yang sia-sia sesuai dengan hasil ujicoba bahwa waktu yang dibutuhkan antara (a) dan (b) relatif sama, namun (c) melakukan pengecekan yang sia-sia sehingga dibutuhkan waktu yang lebih lama.

3. Uji Coba Peningkatan Jumlah Model

Uji coba kali ini menggunakan model HORSE.PLY pada mode gambar terbaik yaitu arsiran penuh. Sebuah model HORSE.PLY membutuhkan waktu render selama 47 milisekon.

Kemudian ditambahkan lagi model yang sama, dimana diperlukan 60 sekon untuk merender dua model tersebut. Begitu seterusnya sampai ditampilkan sepuluh model yang membutuhkan waktu render selama 172 milisekon. Dari grafik yang linier, tampak bahwa peningkatan jumlah model sebanding dengan lama proses yang dibutuhkan.

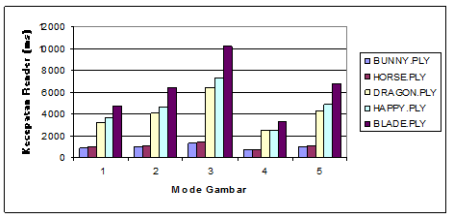


Gambar 13. Grafik Kecepatan Proses Render dan Peningkatan Jumlah Model

4. Uji Coba pada Komputer Lain

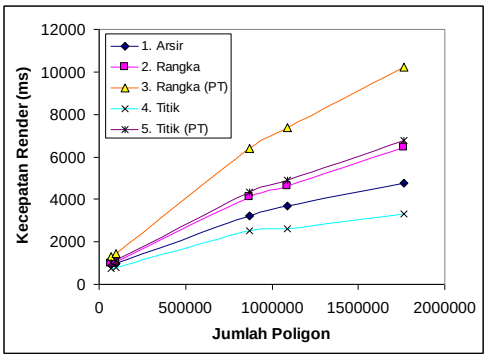
Uji coba kali ini dilakukan pada komputer dengan spesifikasi yang lebih lama, yaitu prosesor Pentium II ® 400 Mhz, memori 128 MB, dan vga card 4MB PCI. Informasi yang didapatkan dari hasil uji coba kecepatan render pada beberapa model adalah sebagai berikut:

Dari informasi diatas diperoleh grafik kecepatan proses render terhadap model uji coba pada mode gambar yang berbeda, sebagai berikut:



Gambar 14. Grafik Kecepatan Proses Render pada Komputer 400 Mhz

Proses render yang paling cepat pada komputer 400 Mhz tanpa kartu pemacu grafis adalah pada mode gambar titik (4), diikuti oleh mode gambar arsir (1), rangka (2), titik tanpa permukaan tersembunyi (5) dan yang paling lama adalah mode gambar rangka tanpa permukaan tersembunyi (3). Pada komputer ini, mode gambar rangka (2) ternyata lebih cepat dibanding mode gambar titik tanpa permukaan tersembunyi (5). Jika dibandingkan dengan hasil uji coba sebelumnya dimana mode gambar 5 lebih cepat dibanding 2, hal ini mungkin disebabkan adanya kartu pemacu grafis pada uji coba sebelumnya yang mempercepat proses pada saat menghilangkan permukaan tersembunyi.



Gambar 15. Grafik Kecepatan Proses Render terhadap Jumlah Poligon pada Komputer 400Mhz

11. KESIMPULAN

Dalam Tugas Akhir ini, telah dilakukan evaluasi terhadap proses perancangan dan pengembangan sistem. Berdasarkan evaluasi tersebut maka dapat diambil beberapa kesimpulan, yaitu :

- Perangkat lunak dapat menampilkan model yang ada dengan sempurna dan dapat dikontrol dengan lancar sesuai fasilitas yang dipilih. Dengan adanya perangkat lunak ini maka visualisasi arsip digital 3D dapat dilakukan pada sebagian besar komputer personal yang ada saat ini dengan menggunakan standar komputer tahun 2003.
- Faktor yang paling dominan dalam mempengaruhi kinerja dari perangkat lunak adalah jumlah seluruh poligon yang terdapat dalam dunia. Namun sudah dilakukan optimisasi yaitu pemangkasan berbasis partisi secara hirarki untuk bagian yang berada diluar batas pandang, tentunya dengan terlebih dahulu melakukan partisi ruang menggunakan metode Octree. Sehingga sesuai dengan hasil uji coba, yang paling berpengaruh adalah jumlah poligon atau partisi yang tampak saja.

12. DAFTAR PUSTAKA

1. OpenGL Architecture Review Board, Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner. 2000, February. OpenGL Programming Guide: the official guide to learning OpenGL, Version 1.2, Third Edition. Silicon Graphics, Inc.
2. Greg Turk. 1994, March. The PLY Polygon File Format. http://www.cc.gatech.edu/ projects/large_models/ply.html
3. Gil Gribb, Klaus Hartmann. 2001, June. Fast Extraction of Viewing Frustum Planes from the World-View-Projection Matrix. http://www2.ravensoft.com/users/ggribb/ plane%20extraction.pdf
4. David Eberly. 2002, March. Intersection of a Sphere and a Cone. Magic Software, Inc. http://www.magic-software.com
5. Mike Kelleghan. 1997, July. Octree Partitioning Techniques. http://www.gamasutra.com/features/19970801/octree.htm
6. DeLaura, M. 2001, March. Game Programming Gems, Volume 1. Charles River Media.
7. Alan Watt, Fabio Policarpo. 2000, December. 3D Games: Real-time Rendering and Software Technology, Volume 1. Addison-Wesley.